AMENDMENTS TO THE CLAIMS

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A computer-implemented compressed-code generating method that is used for compressing information, ~~including letters, numbers, sound or images,~~ the method comprising:

receiving a binary code bit string {y} which represents data, sound or images;

~~a first step of~~ obtaining first and second bit strings $\{y\}_1$ and $\{y\}_2$ respectively from ~~a~~ the binary code bit string {y}~~, which represents information to be compressed~~;

~~a second step of~~ defining a quantized initial value Y(0) by giving a binary weight to the first bit string~~s~~ $\{y\}_1$;

~~a third step of~~ obtaining an internal status x(0) using the quantized initial value Y(0) and **n** in a right-hand side of an inverse ~~conversion~~ transform of ~~in-phase conversion~~ isomorphic transform and quantization expression, $x(0) = \{\sin \pi Y_n(0) / 2^{n+1}\}^2$ (where **n** is a quantized resolution of the first bit string);

~~a fourth step of~~ substituting the obtained internal status x(0) in a right-hand side of an inverse calculation expression of a logistic map, $x(t-1) = (1 \pm (1-x(t))^{1/2})/2$ (where **t** is a discrete time), and selecting a sign of positive or negative in the right-hand side of the inverse calculation expression according to a binary value of the second bit string $\{y\}_2$, thereby obtaining a past retroactive internal status, ~~the fourth step being~~ the substituting and selecting steps sequentially executed by the number of bits of the second strings $\{y\}_2$; ~~and~~

~~a fifth step of~~ generating a compressed code $Y(-\tau)$ by using a resultant internal status $x(-\tau)$ ~~obtained at the fourth step~~ and **m** in a right-hand side of an ~~in-phase conversion~~ isomorphic transform and quantization expression, $Y(-\tau) = 2/\pi \times \arcsin(x(-\tau))^{1/2} \times 2^m$ (where

m is a newly defined quantized resolution of the second bit string ~~newly defined for the above n~~, and τ is a discrete time newly defined for the above t)~~,~~; and

outputting the compressed code $Y(-\tau)$,

wherein a length of the compressed code $Y(-\tau)$ is less than a length of the bit string {y}.

2. (Currently amended) A computer-implemented compressed-code expanding method that is used for restoring and expanding the compressed code $Y(-\tau)$ generated by using the compressed-code generating method recited in Claim 1, the method comprising:

receiving the compressed code $Y(-\tau)$;

~~a sixth step of~~ obtaining the internal status $x(-\tau)$ using the compressed code $Y(-\tau)$ and n in an inverse ~~conversion~~ transform ~~expression~~ of ~~in phase conversion~~ isomorphic transform and quantization, $x(-\tau) = \{\sin \pi Y(-\tau) / 2^{n+1}\}^2$;

~~a seventh step of~~ obtaining an internal status using the internal status $x(-\tau)$ in a right-hand side of forward calculation expressions of a logistic map

$$x(t+1) = 4x(t)\{1 - x(t)\} \quad \ldots(1)$$

$$x(t) = x(t + 1) \quad \ldots(2)$$

~~the seventh step being~~ which is executed repeatedly up to an internal status x(0);

~~an eighth step of~~ restoring and expanding the second bit string $\{y\}_2$ by sequentially using the internal status x(t) ~~obtained at the seventh step~~ and m in a right-hand side of an ~~in-phase conversion~~ isomorphic transform and quantization expression, $Y(t) = 2/\pi \times \arcsin(x(t))^{1/2} \times 2^m$; ~~and~~

~~a ninth step of~~ restoring and expanding the first bit string $\{y\}_1$ by sequentially using the internal status x(0) ~~obtained at the seventh step~~ and m in the right-hand side of the ~~in-phase conversion~~ isomorphic transform and quantization expression; and

outputting the restored first bit string $\{y\}_1$ and second bit string $\{y\}_2$.

3. (Previously presented) The computer-implemented compressed-code generating method of claim 1 wherein the sign of the right-hand side of the inverse calculation expression of the logistic map is positive when the value of the second bit string is 1 and negative when the value is 0.